

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-182931

(43)Date of publication of application : 28.06.2002

(51)Int.Cl.

G06F 9/46

(21)Application number : 2000-384081

(71)Applicant : YASKAWA ELECTRIC CORP

(22)Date of filing : 18.12.2000

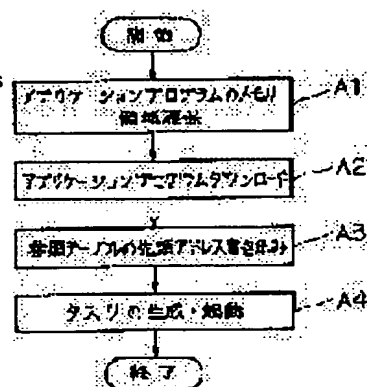
(72)Inventor : YUMIBA TADASUKE

(54) COMMON OS SYSTEM CALL METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To enable development of an application program not to depend on an OS.

SOLUTION: A common OS system call capable of selectively calling a system call of an operating system counted on an incorporating device is generated from system calls of several kinds of operating systems and a reference table to define starting addresses of the respective common OS system calls as an element is generated. The application program can indirectly call the system call of the operating system mounted on the incorporating device by calling the common OS system call by using the reference table.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-182931

(P2002-182931A)

(43) 公開日 平成14年6月28日 (2002.6.28)

(51) Int.Cl.⁷

G 0 6 F 9/46

識別記号

3 4 0

F I

G 0 6 F 9/46

テーム* (参考)

3 4 0 A 5 B 0 9 8

審査請求 未請求 請求項の数 3 O L (全 5 頁)

(21) 出願番号 特願2000-384081 (P2000-384081)

(22) 出願日 平成12年12月18日 (2000.12.18)

(71) 出願人 060006622

株式会社安川電機

福岡県北九州市八幡西区黒崎城石2番1号

(72) 発明者 弓場 忠輔

福岡県北九州市八幡西区黒崎城石2番1号

株式会社安川電機内

(74) 代理人 100088328

弁理士 金田 暢之 (外2名)

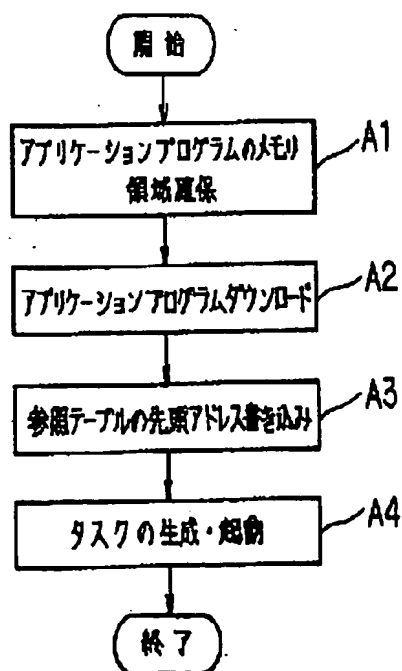
Fターム(参考) 5B098 BA06 EE02 GA01 GA02 GB01

(54) 【発明の名称】 共通OSシステムコール方法

(57) 【要約】

【課題】 OSに依存しないアプリケーションプログラムの開発を可能とする。

【解決手段】 数種類のオペレーティングシステムのシステムコールの中から組み込み装置に実装されているオペレーティングシステムのシステムコールを選択的に呼び出し可能な共通OSシステムコールを作成し、各共通OSシステムコールの開始アドレスを要素とする参照テーブルを作成する。アプリケーションプログラムは、参照テーブルを用いて共通OSシステムコールを呼び出すことによって、組み込み装置に実装されているオペレーティングシステムのシステムコールを間接的に呼び出すことができる。



【特許請求の範囲】

【請求項1】 数種類のオペレーティングシステムのシステムコールの中から実際に組み込み装置に実装されているオペレーティングシステムのシステムコールを選択的に呼び出し可能な共通OSシステムコールを作成し、前記各共通OSシステムコールの開始アドレスを要素とする参照テーブルを作成し、

前記参照テーブルを用いて前記共通OSシステムコールを呼び出すことによって、前記組み込み装置に実装されているオペレーティングシステムのシステムコールを間接的に呼び出す共通OSシステムコール方法。

【請求項2】 前記共通OSシステムコールは、アプリケーションプログラムから直接呼び出される請求項1記載の共通OSシステムコール方法。

【請求項3】 前記共通OSシステムコールは、アプリケーションプログラムと前記オペレーティングシステムとの間のインタフェースであるアプリケーションプログラムインタフェースによって呼び出され、前記アプリケーションプログラムは、前記アプリケーションプログラムインタフェースを呼び出すことによって前記共通OSシステムコールの間接呼び出しを行なう請求項1記載の共通OSシステムコール方法。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、ダウンロードされたアプリケーションプログラムを実行する組み込み型装置に実装されたオペレーティングシステム（以下、OS）のシステムコールを呼び出すための共通OSシステムコール方法に関する。

【0002】

【従来の技術】 従来、ダウンロードされたアプリケーションプログラムインタフェース（以下、API）を使用しないアプリケーションプログラムを実行する組み込み型装置では、アプリケーションプログラムが直接、OSシステムコールを呼び出していた。

【0003】 図5には、OSシステムコールの一例が示されている。これらのOSシステムコールは、C言語で記載されている。図5(a)の関数GetMemory()と図5(b)の関数AllocMemory()とは、それぞれOS_AとOS_Bとにおけるメモリを確保するためのシステムコールである。

【0004】 図5に示すように、OSが異なると、システムコールの関数名および引数は異なったものになる。そのため、従来の組み込み装置では、OSが修正あるいは変更された場合に、システムコールを呼び出すアプリケーションプログラムのソースコードを書き直さなければならなくなるという問題があった。

【0005】 また、直接システムコールを呼び出すようなアプリケーションプログラムを開発するには、OS等のシステムプログラムに関する深い知識が必要となるた

め、アプリケーションプログラムの開発が容易でないという問題もあった。

【0006】

【発明が解決しようとする課題】 以上述べたように、従来の組み込み型装置では、以下に示す2つの問題点を有している。

(1) OSが修正あるいは変更された場合に、システムコールを呼び出すアプリケーションプログラムのソースコードを書き直さなければならなくなる。

(2) アプリケーションプログラムの開発においても、OS等のシステムプログラムに関する深い知識が必要となり、アプリケーションプログラムの開発が容易でない。

【0007】 本発明は、OSに依存しないアプリケーションプログラムの開発を可能とする共通OSシステムコール方法を提供することを目的とする。

【0008】 また、本発明は、容易にアプリケーションプログラムの開発することができる共通OSシステムコール方法を提供することを目的とする。

【0009】

【課題を解決するための手段】 上記課題を解決するために、本発明の共通OSシステムコール方法では、数種類のオペレーティングシステムのシステムコールの中から実際に組み込み装置に実装されているオペレーティングシステムのシステムコールを選択的に呼び出し可能な共通OSシステムコールを作成し、前記各共通OSシステムコールの開始アドレスを要素とする参照テーブルを作成し、前記参照テーブルを用いて前記共通OSシステムコールを呼び出すことによって、前記組み込み装置に実装されているオペレーティングシステムのシステムコールを間接的に呼び出す。

【0010】 本発明の共通OSシステムコール方法では、いずれのオペレーティングシステムが組み込み型装置に実装されていても、そのオペレーティングシステムのシステムコールを呼び出し可能な共通OSシステムコールを用いてアプリケーションプログラムの開発を行なうことができるため、OSに依存しないアプリケーションプログラムの開発が可能となる。

【0011】 本発明の他の共通OSシステムコール方法では、前記共通OSシステムコールは、アプリケーションプログラムと前記オペレーティングシステムとの間のインタフェースであるアプリケーションプログラムインタフェースによって呼び出され、前記アプリケーションプログラムは、前記アプリケーションプログラムインタフェースを呼び出すことによって前記共通OSシステムコールの間接呼び出しを行なう。

【0012】 本発明の共通OSシステムコール方法では、アプリケーションプログラムインタフェースが共通OSシステムコールの呼び出しを行なうので、アプリケーションプログラムの開発者は、OS等のシステムプログ

ラムに関する深い知識を持たなくてもアプリケーションの開発を行なうことができるようになるため、容易にアプリケーションプログラムの開発することができるようになる。

【0013】

【発明の実施の形態】次に、本発明の一実施形態の共通OSシステムコール方法について図面を参照して詳細に説明する。本実施形態の共通OSシステムコール方法では、OSがOS_Aであっても、OS_Bであっても呼び出し可能な共通OSシステムコールとしての関数を用意する。図1には、本実施形態の共通OSシステムコール方法における共通OSシステムコールの一例が示されている。

【0014】図1の関数`ye_get_mem 0`は、OS_AおよびOS_Bに共通なシステムコールである。この関数`ye_get_mem 0`では、もし、“OS_A”が真であれば、OS_Aのシステムコール`GetMemory()`が実行され、“OS_A”が偽であればOS_Bのシステムコール`AllocMemory()`が実行される。つまり、共通OSシステムコールは、数種類のOSのシステムコールの中から実際に組み込み装置に実装されているOSのシステムコールを選択的に呼び出し可能となっている。このような共通OSシステムコールを用意すれば、従来のように、OSを変更する毎にアプリケーションプログラムのソースコードも変更する必要がなくなる。

【0015】本実施形態の共通OSシステムコール方法では、上述のような共通OSシステムコールを数十個用意し、図2に示す参照テーブルを作成する。この参照テーブルの要素は各共通OSシステムコールの開始アドレス、すなわち各共通OSシステムコールへのポインタとなっている。

【0016】図3は、本実施形態の共通OSシステムコール方法を用いて作成されたアプリケーションプログラムが組み込み装置のメモリにダウンロードされた際のメモリ配置を示す図である。アプリケーションプログラムのメモリ配置は、ヘッダセクション1と、スタートアップ関数2と、テキストセクション3と、データセクション4とから構成される。

【0017】ヘッダセクション1の固定オフセットアドレスには、前述の参照テーブルが書き込まれる。スタートアップ関数2はアプリケーションプログラム生成時にアプリケーションプログラムにリンクされる関数であり、その処理の先頭アドレスは、アプリケーションプログラムの実行が開始されるときの実行開始アドレスとなっており、その実行が開始されるときに最初に実行される関数である。

【0018】スタートアップ関数2は、アプリケーションプログラムのタスクの起動や、組み込み装置のローカルメモリへの参照テーブルの読み出し等の処理を行な

う。テキストセクション3には、アプリケーションプログラムのコードが格納され、データセクション4には、アプリケーションプログラムのデータが格納される。

【0019】このアプリケーションプログラムにおいて、共通OSシステムコールが呼ばれると、組み込み装置のCPUは、まず、ヘッダセクション1の固定オフセットアドレスに格納されている参照テーブルへのポインタに基づいて参照テーブルへアクセスし、参照テーブルに格納されている呼び出された共通OSシステムコールへのポインタに基づいて、共通OSシステムコールにジャンプする。

【0020】図4は、本実施形態の共通OSシステムコール方法におけるアプリケーションプログラムをダウンロードする際の動作を示すフローチャートである。まず、アプリケーションプログラムを書き込むのに必要な領域を、組み込み型装置内部のメモリから確保し（ステップA1）、確保された領域にアプリケーションプログラムをダウンロードする（ステップA2）。次に、図5に示すヘッダセクション1内の固定オフセットアドレスに、参照テーブルへのポインタとして参照テーブルの先頭アドレスを書き込む（ステップA3）。そして、スタートアップ関数2を実行してタスクの生成および起動を行なう（ステップA4）。

【0021】以上述べたように、本実施形態の共通OSシステムコール方法では、いずれのオペレーティングシステムが前記組み込み型装置に実装されていても、当該オペレーティングシステムのシステムコールを呼び出し可能な共通OSシステムコールを用いてアプリケーションプログラムの開発を行なうことができるため、OSに依存しないアプリケーションプログラムの開発が可能となる。

【0022】また、本実施形態の共通OSシステムコール方法では、アプリケーションプログラムが直接、共通OSシステムコールの呼出しを行なったが、本発明の共通OSシステムコール方法はこれに限定されるものではなく、アプリケーションプログラムが直接、共通OSシステムコールの呼出しを行わずに、アプリケーションプログラムインタフェース（以下、API）が共通OSシステムコールの呼出しを行なうようにしてもよい。図6にAPIが共通OSシステムコールの呼出しを行う場合の一例を示す。図6(a)はアプリケーションのソースコードであり、図6(b)はAPIのソースコードである。図6(a)に示されるアプリケーションプログラムは、APIの関数、例えば、図6(b)の`yeAllocMemory()`を読み出すことによって組み込み型装置に対する指令を実行する。APIの関数、例えば`yeAllocMemory()`は、アプリケーションプログラムからの指令に応じ、必要に応じて共通OSシステムコールの間接呼び出しを行なう。

【0023】この場合、アプリケーションプログラムイ

ンタフェースが共通OSシステムコールの呼出しを行なうので、アプリケーションプログラムの開発者は、OS等のシステムプログラムに関する深い知識を持たなくてもアプリケーションの開発を行なうことができるようになるため、容易にアプリケーションプログラムの開発することができるようになる。

【0024】

【発明の効果】以上述べたように、本発明の共通OSシステムコール方法は、以下に示す2つの利点を有する。

(1) いずれのオペレーティングシステムが組み込み型装置に実装されていても、当該オペレーティングシステムのシステムコールを呼び出し可能な共通OSシステムコールを用いてアプリケーションプログラムの開発を行なうことができるため、OSに依存しないアプリケーションプログラムの開発が可能となる。

(2) アプリケーションプログラムインタフェースが共通OSシステムコールの呼出しを行なうので、アプリケーションプログラムの開発者は、OS等のシステムプログラムに関する深い知識を持たなくてもアプリケーションプログラムの開発を行なうことができるようになるため、容易にアプリケーションプログラムの開発することができるようになる。

【図面の簡単な説明】

【図1】 本発明の一実施形態の共通OSシステムコール方法における共通OSシステムコールの一例を示す図である。

【図2】 本発明の一実施形態の共通OSシステムコール方法における参照テーブルの一例を示す図である。

【図3】 本発明の一実施形態の共通OSシステムコール方法を用いて作成されたアプリケーションプログラムが組み込み装置のメモリにダウンロードされた際のメモリ配置を示す図である。

【図4】 本発明の一実施形態の共通OSシステムコール方法におけるアプリケーションプログラムをダウンロードする際の動作を示すフローチャートである。

【図5】 OSシステムコールの一例を示す図である。

【図6】 APIが共通OSシステムコールの呼出しを行なう場合の一例を示す図である。

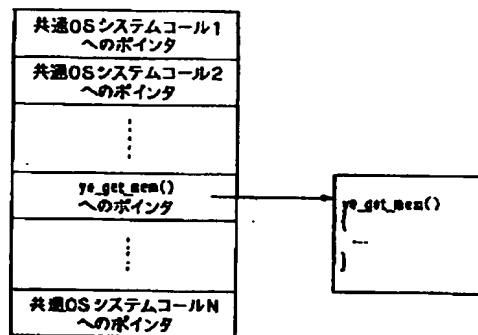
【符号の説明】

- 1 ヘッドセクション
- 2 スタートアップ関数
- 3 テキストセクション
- 4 データセクション
- A1～A4 ステップ

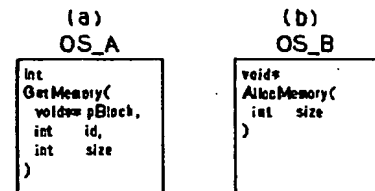
【図1】

```
int
ye_get_mem(
void** pBlock,
int size,
)
{
#if OS_A
GetMemory();
#else
AllocMemory();
#endif
return;
}
```

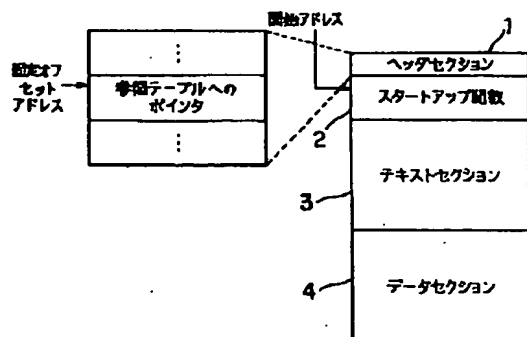
【図2】



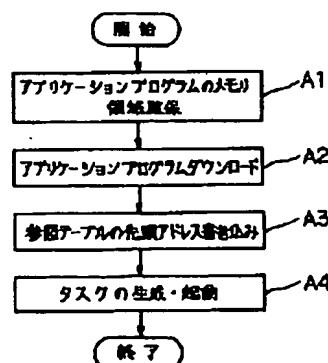
【図5】



【図3】



【図4】



【図6】

(a)
アプリケーション
プログラム

```
void  
main(  
    int argc,  
    char **argv,  
)  
{  
    int size;  
    char *pBlock;  
    ...  
    pBlock=yeAllocMemory(size);  
    ...  
    return;  
}
```

(b)

API

```
char*  
yeAllocMemory(  
    int size  
)  
{  
    ...  
    共通OSシステムコールの  
    呼び出し  
    ...  
}
```